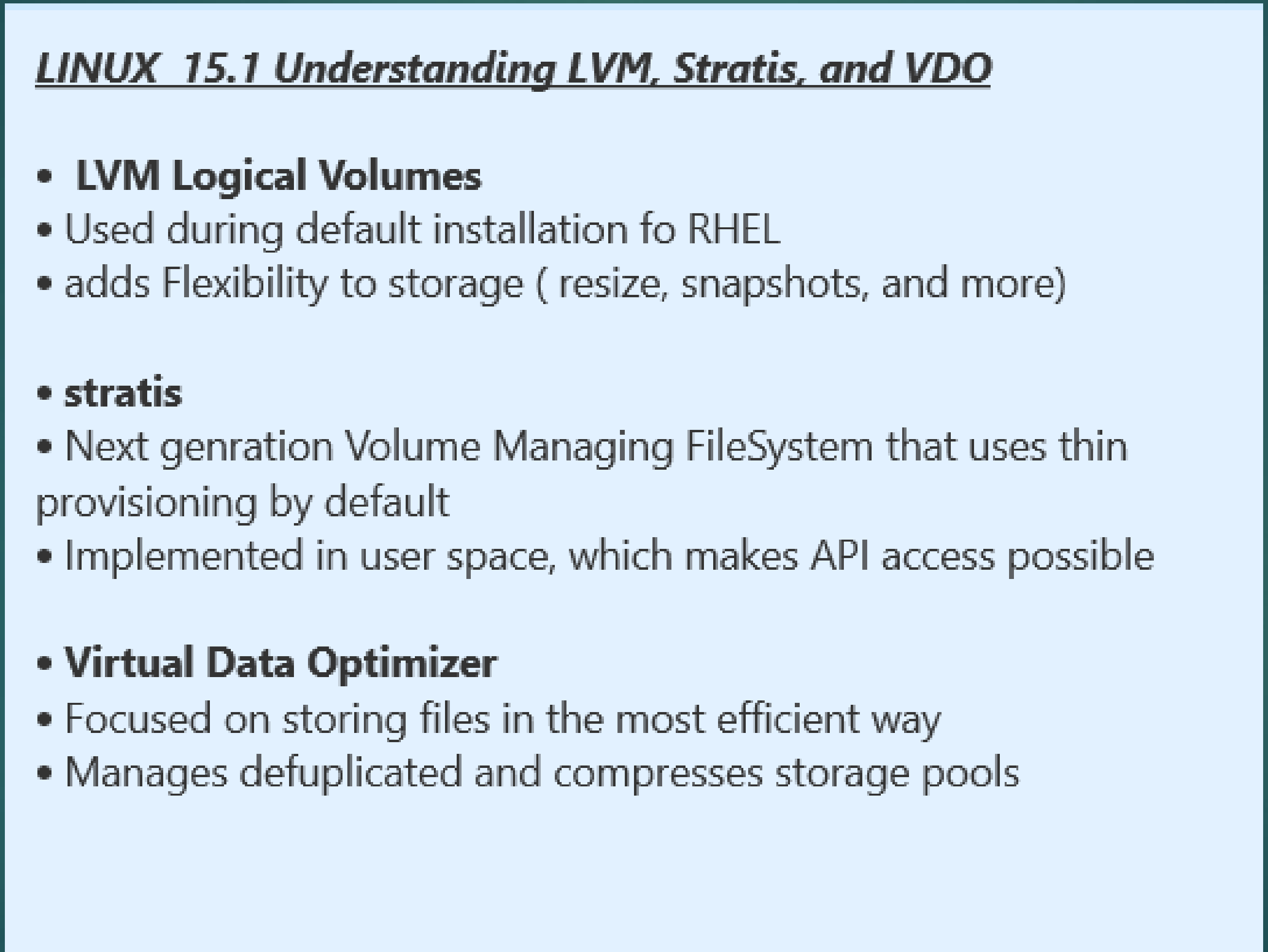
# Chapter 15

BY:

ETISHA JAIN

VISHU GOYAL

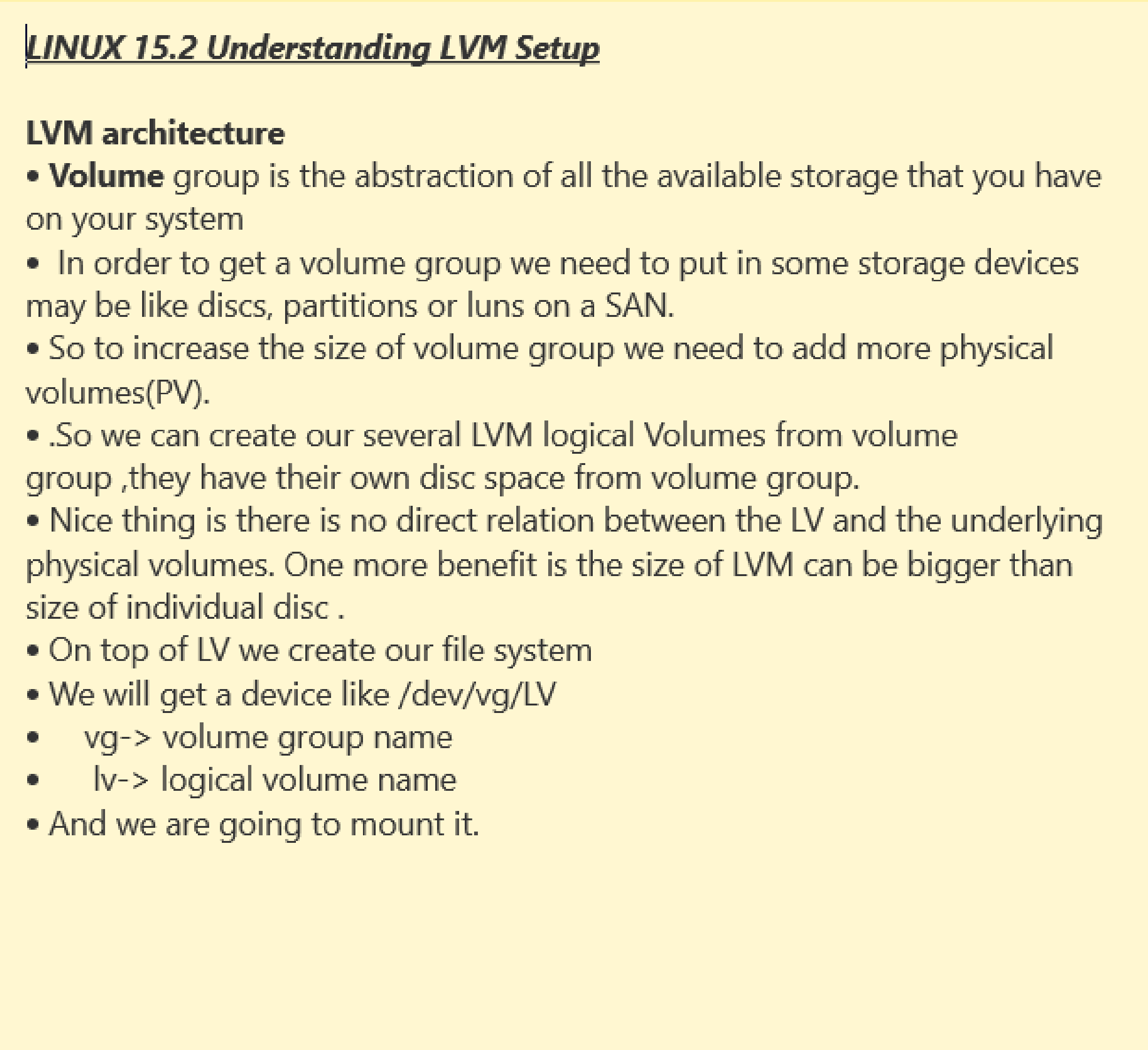


## ***LINUX  15.1 Understanding LVM, Stratis, and VDO***

- **LVM Logical Volumes**
- Used during default installation fo RHEL
- adds Flexibility to storage ( resize, snapshots, and more)

- **stratis**
- Next genration Volume Managing FileSystem that uses thin provisioning by default
- Implemented in user space, which makes API access possible

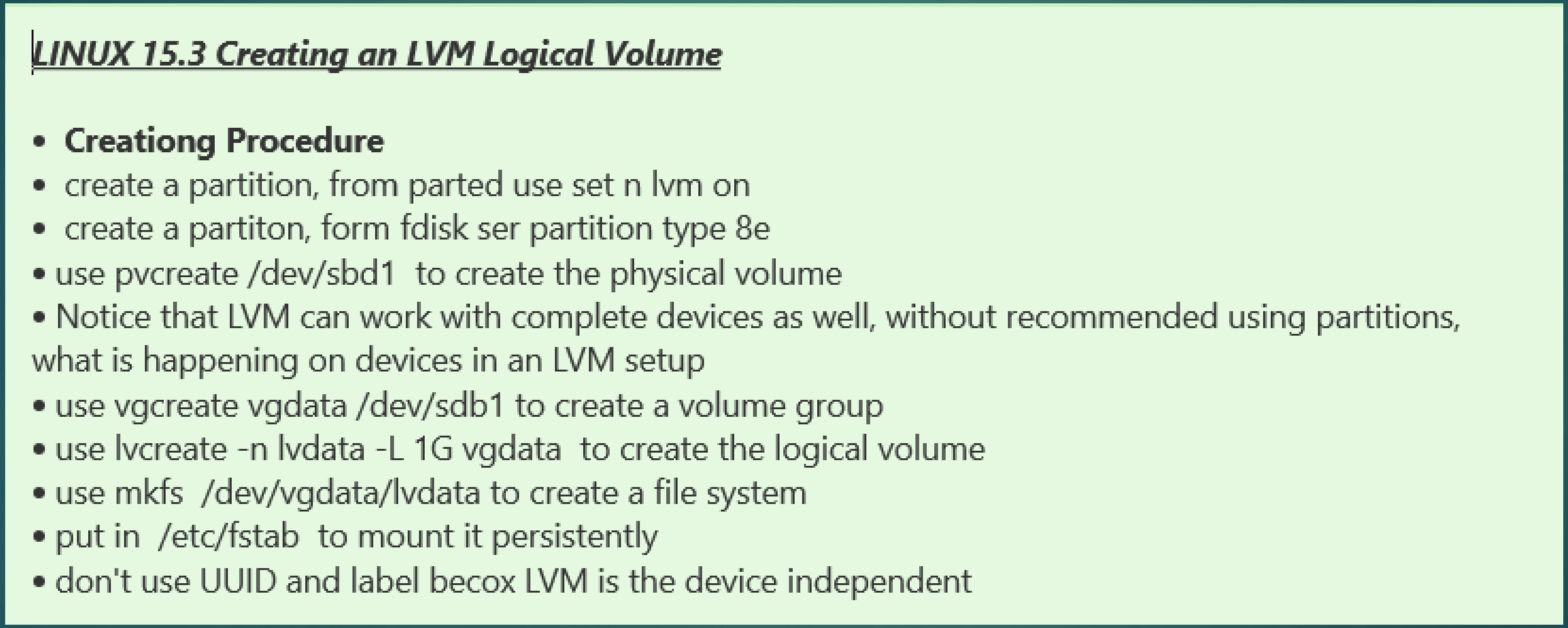
- **Virtual Data Optimizer**
- Focused on storing files in the most efficient way
- Manages defuplicated and compresses storage pools



***LINUX 15.2 Understanding LVM Setup***

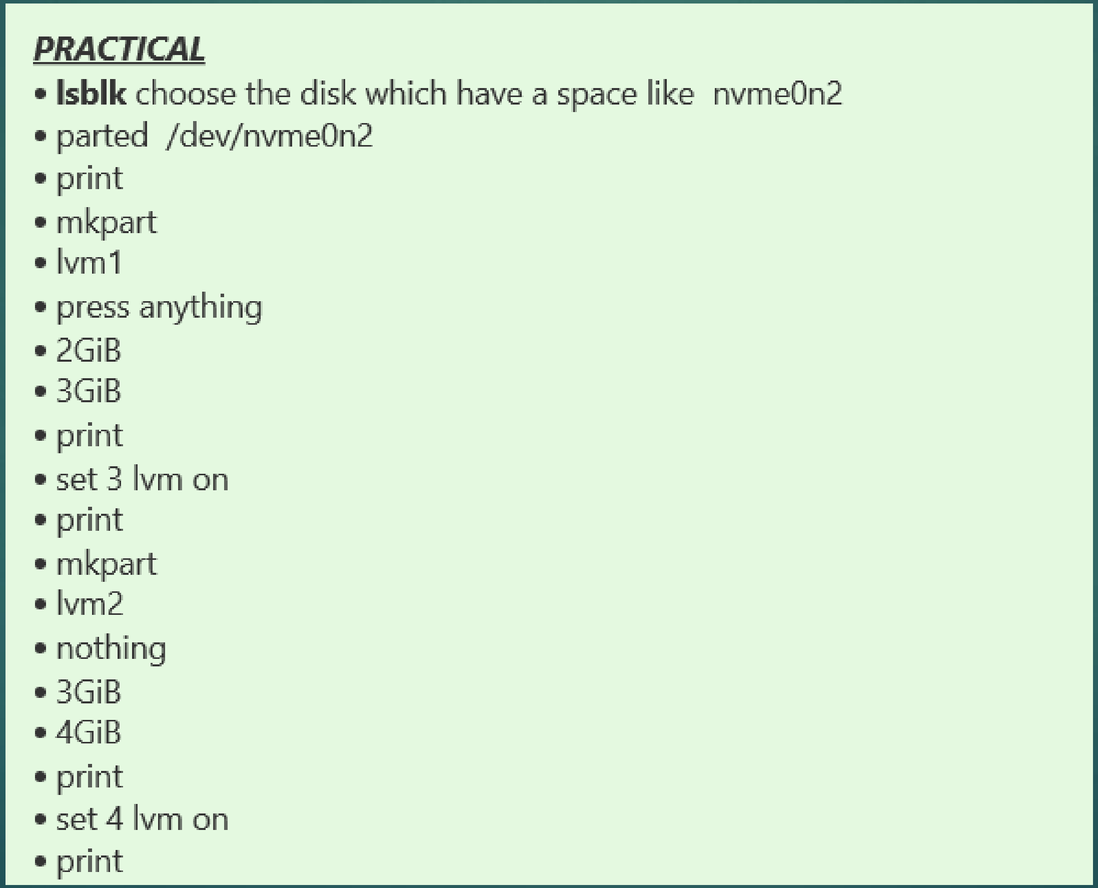
**LVM architecture**

- **Volume** group is the abstraction of all the available storage that you have on your system
- In order to get a volume group we need to put in some storage devices may be like discs, partitions or luns on a SAN.
- So to increase the size of volume group we need to add more physical volumes(PV).
- .So we can create our several LVM logical Volumes from volume group ,they have their own disc space from volume group.
- Nice thing is there is no direct relation between the LV and the underlying physical volumes. One more benefit is the size of LVM can be bigger than size of individual disc .
- On top of LV we create our file system
- We will get a device like /dev/vg/LV
- vg-> volume group name
- lv-> logical volume name
- And we are going to mount it.



## *LINUX 15.3 Creating an LVM Logical Volume*

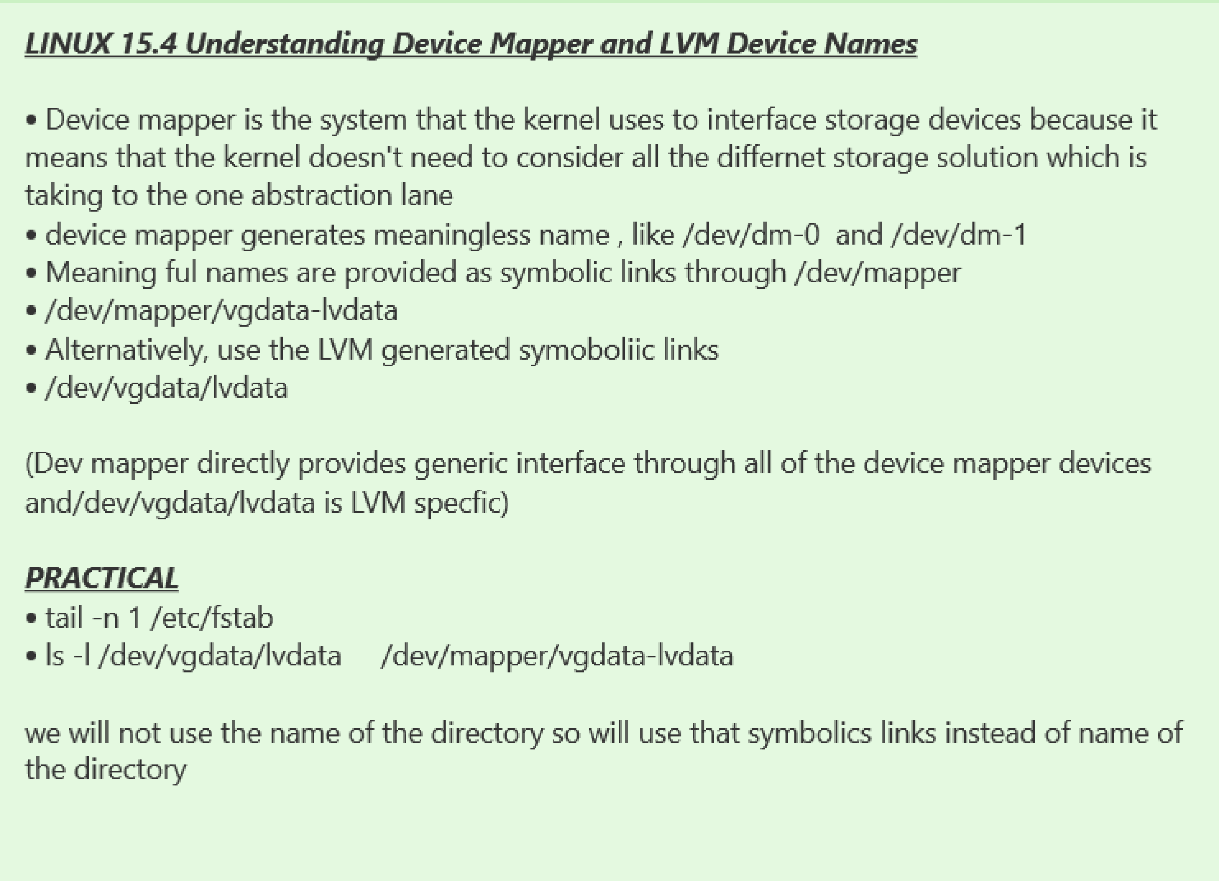
- **Creationg Procedure**
- create a partition, from parted use set n lvm on
- create a partiton, form fdisk ser partition type 8e
- use pvcreate /dev/sbd1 to create the physical volume
- Notice that LVM can work with complete devices as well, without recommended using partitions, what is happening on devices in an LVM setup
- use vgcreate vgdata /dev/sdb1 to create a volume group
- use lvcreate -n lvdata -L 1G vgdata to create the logical volume
- use mkfs /dev/vgdata/lvdata to create a file system
- put in /etc/fstab to mount it persistently
- don't use UUID and label becox LVM is the device independent



***PRACTICAL***

- **lsblk** choose the disk which have a space like  nvme0n2
- parted  /dev/nvme0n2
- print
- mkpart
- lvm1
- press anything
- 2GiB
- 3GiB
- print
- set 3 lvm on
- print
- mkpart
- lvm2
- nothing
- 3GiB
- 4GiB
- print
- set 4 lvm on
- print

- print
- pvcreate  /dev/nvme0np3
- device not found
- lsblk
- pvcreate  /dev/nvme0n2p3
- successfully
- pvs to verify the existing of the physical volume
- if VG group is empty then it is sure that we havnt assign that partition till now
- vgcreate vgdata /dev/nvme0n2p3
- pvs
- vgs
- lvcreate  -n  lvdata  -L 812M  vgdata
- lvs
- man lvcreate
- mkfs.xfs  /dev/vgdata/lvdata
- mkdire /mounts/lvm1
- vim  /etc/fstab
- /dev/vgdta/lvdata       /mounts/lvm1    xfs   defaults   0   0
- mount -a
- mount
- findmnt



### *LINUX 15.4 Understanding Device Mapper and LVM Device Names*

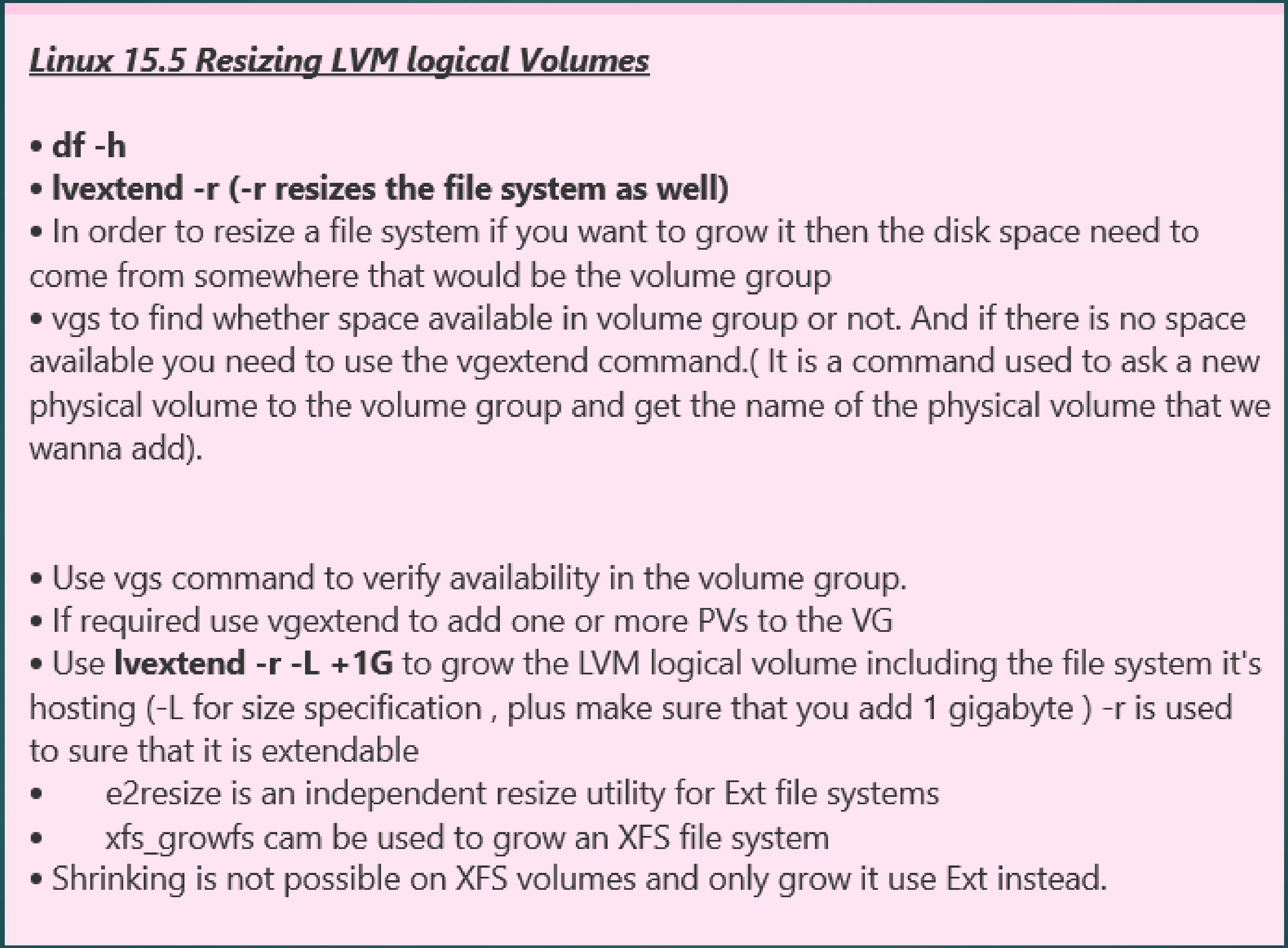
- Device mapper is the system that the kernel uses to interface storage devices because it means that the kernel doesn't need to consider all the differnet storage solution which is taking to the one abstraction lane
- device mapper generates meaningless name , like /dev/dm-0  and /dev/dm-1
- Meaning ful names are provided as symbolic links through /dev/mapper
- /dev/mapper/vgdata-lvdata
- Alternatively, use the LVM generated symoboliic links
- /dev/vgdata/lvdata

(Dev mapper directly provides generic interface through all of the device mapper devices and/dev/vgdata/lvdata is LVM specfic)

***PRACTICAL***

- tail -n 1 /etc/fstab
- ls -l /dev/vgdata/lvdata     /dev/mapper/vgdata-lvdata

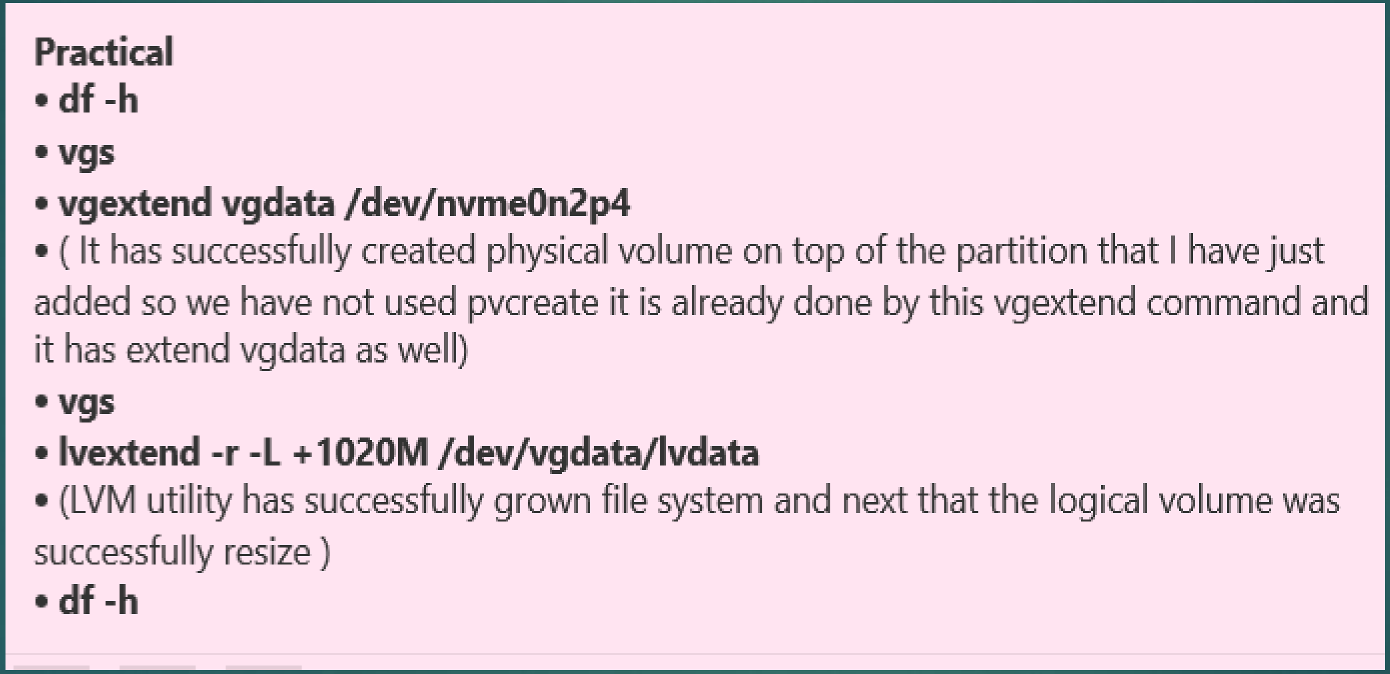
we will not use the name of the directory so will use that symbolics links instead of name of the directory



## ***Linux 15.5 Resizing LVM logical Volumes***

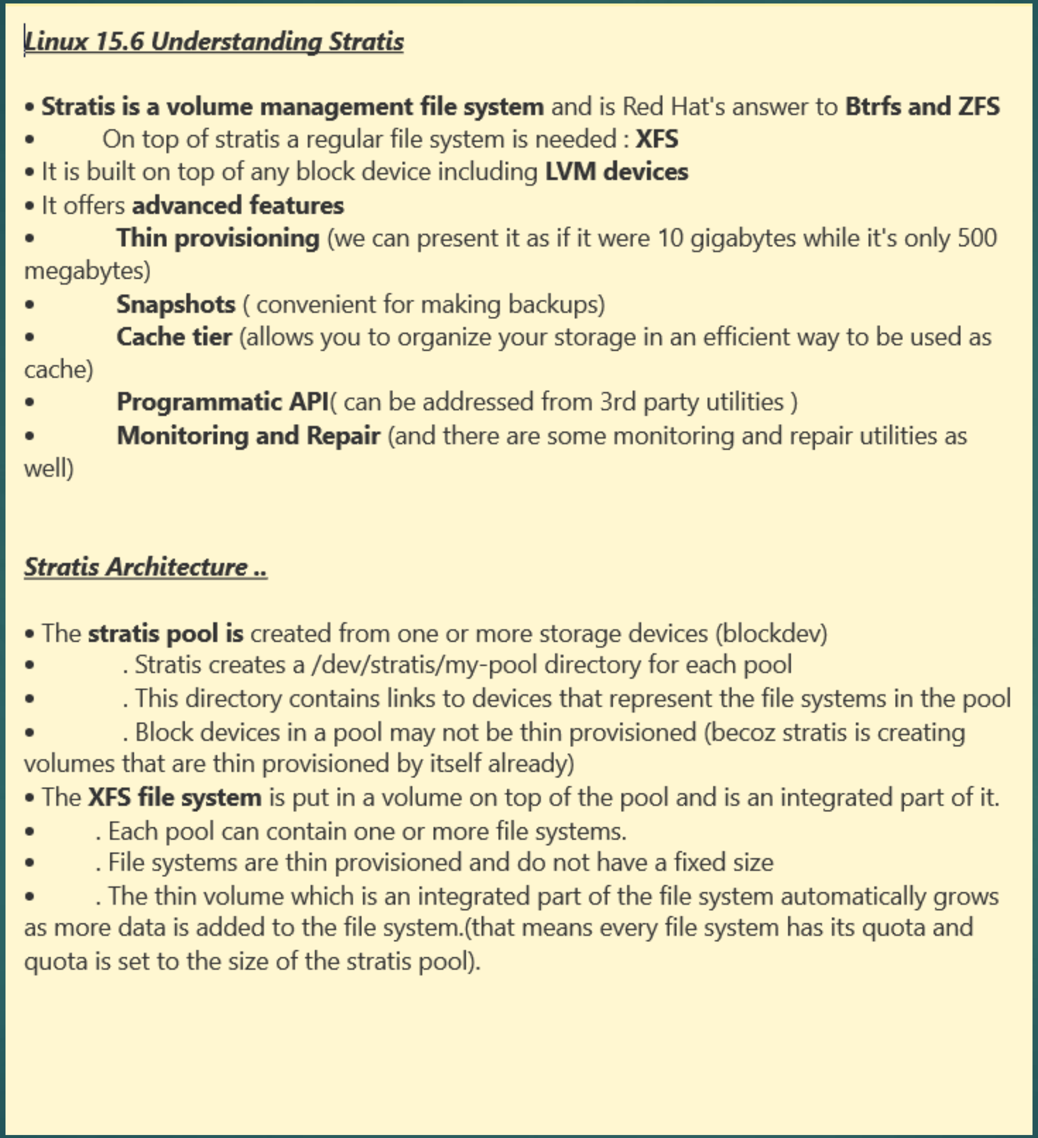
- **df -h**
- **lvextend -r (-r resizes the file system as well)**
- In order to resize a file system if you want to grow it then the disk space need to come from somewhere that would be the volume group
- vgs to find whether space available in volume group or not. And if there is no space available you need to use the vgextend command.( It is a command used to ask a new physical volume to the volume group and get the name of the physical volume that we wanna add).

- Use vgs command to verify availability in the volume group.
- If required use vgextend to add one or more PVs to the VG
- Use **lvextend -r -L +1G** to grow the LVM logical volume including the file system it's hosting (-L for size specification , plus make sure that you add 1 gigabyte ) -r is used to sure that it is extendable
- e2resize is an independent resize utility for Ext file systems
- xfs_growfs cam be used to grow an XFS file system
- Shrinking is not possible on XFS volumes and only grow it use Ext instead.



**Practical**

- **df -h**
- **vgs**
- **vgextend vgdata /dev/nvme0n2p4**
- ( It has successfully created physical volume on top of the partition that I have just added so we have not used pvcreate it is already done by this vgextend command and it has extend vgdata as well)
- **vgs**
- **lvextend -r -L +1020M /dev/vgdata/lvdata**
- (LVM utility has successfully grown file system and next that the logical volume was successfully resize )
- **df -h**

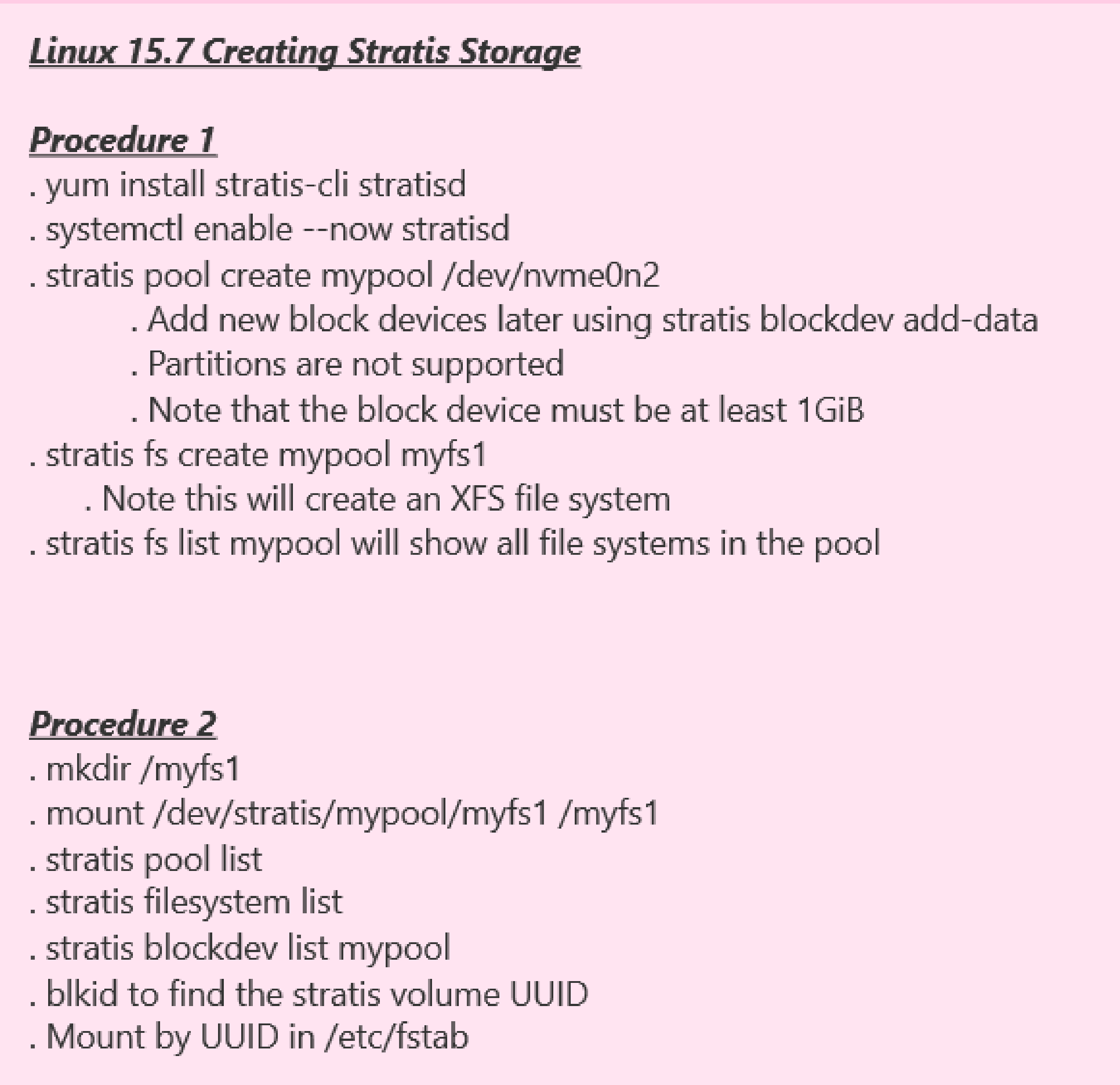


***Linux 15.6 Understanding Stratis***

- **Stratis is a volume management file system** and is Red Hat's answer to **Btrfs and ZFS**
- On top of stratis a regular file system is needed : **XFS**
- It is built on top of any block device including **LVM devices**
- It offers **advanced features**
- **Thin provisioning** (we can present it as if it were 10 gigabytes while it's only 500 megabytes)
- **Snapshots** ( convenient for making backups)
- **Cache tier** (allows you to organize your storage in an efficient way to be used as cache)
- **Programmatic API**( can be addressed from 3rd party utilities )
- **Monitoring and Repair** (and there are some monitoring and repair utilities as well)

***Stratis Architecture ..***

- The **stratis pool is** created from one or more storage devices (blockdev)
- . Stratis creates a /dev/stratis/my-pool directory for each pool
- . This directory contains links to devices that represent the file systems in the pool
- . Block devices in a pool may not be thin provisioned (becoz stratis is creating volumes that are thin provisioned by itself already)
- The **XFS file system** is put in a volume on top of the pool and is an integrated part of it.
- . Each pool can contain one or more file systems.
- . File systems are thin provisioned and do not have a fixed size
- . The thin volume which is an integrated part of the file system automatically grows as more data is added to the file system.(that means every file system has its quota and quota is set to the size of the stratis pool).



## ***Linux 15.7 Creating Stratis Storage***

### ***Procedure 1***

. yum install stratis-cli stratisd

. systemctl enable --now stratisd

. stratis pool create mypool /dev/nvme0n2

. Add new block devices later using stratis blockdev add-data

. Partitions are not supported

. Note that the block device must be at least 1GiB

. stratis fs create mypool myfs1

. Note this will create an XFS file system

. stratis fs list mypool will show all file systems in the pool

### ***Procedure 2***

. mkdir /myfs1

. mount /dev/stratis/mypool/myfs1 /myfs1

. stratis pool list

. stratis filesystem list

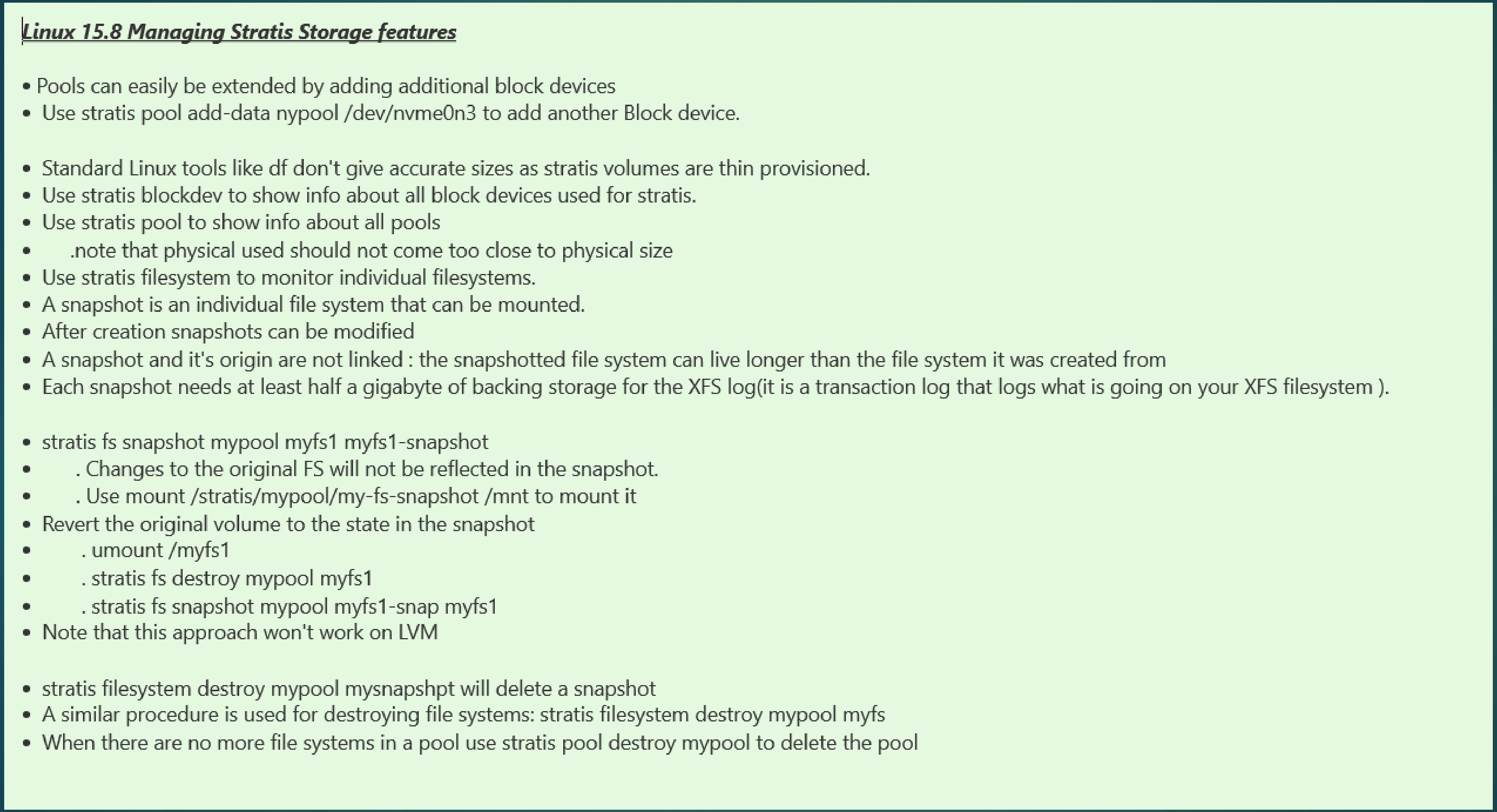
. stratis blockdev list mypool

. blkid to find the stratis volume UUID

. Mount by UUID in /etc/fstab

***PRACTICLE***

```
- systemctl enable --now stratisd
- stratis pool create mypool /dev/nvme0n2
- stratis pool
- stratis fs create mypool myfs1
- stratis fs
- mkdir /myfs1
- mount | grep myfs1
- mount /dev/stratis/mypool/myfs1 /myfs1
- mount
- stratis pool list
- stratis fs list
- stratis blockdev list mypool
- lsblk
- blkid
Copy UUID
- vim /etc/fstab
  UUID=" "   /myfs1 xfs defaults 0 0
- reboot
- su -
- mount | grep myfs
(Mount stratis pool using UUID)
```



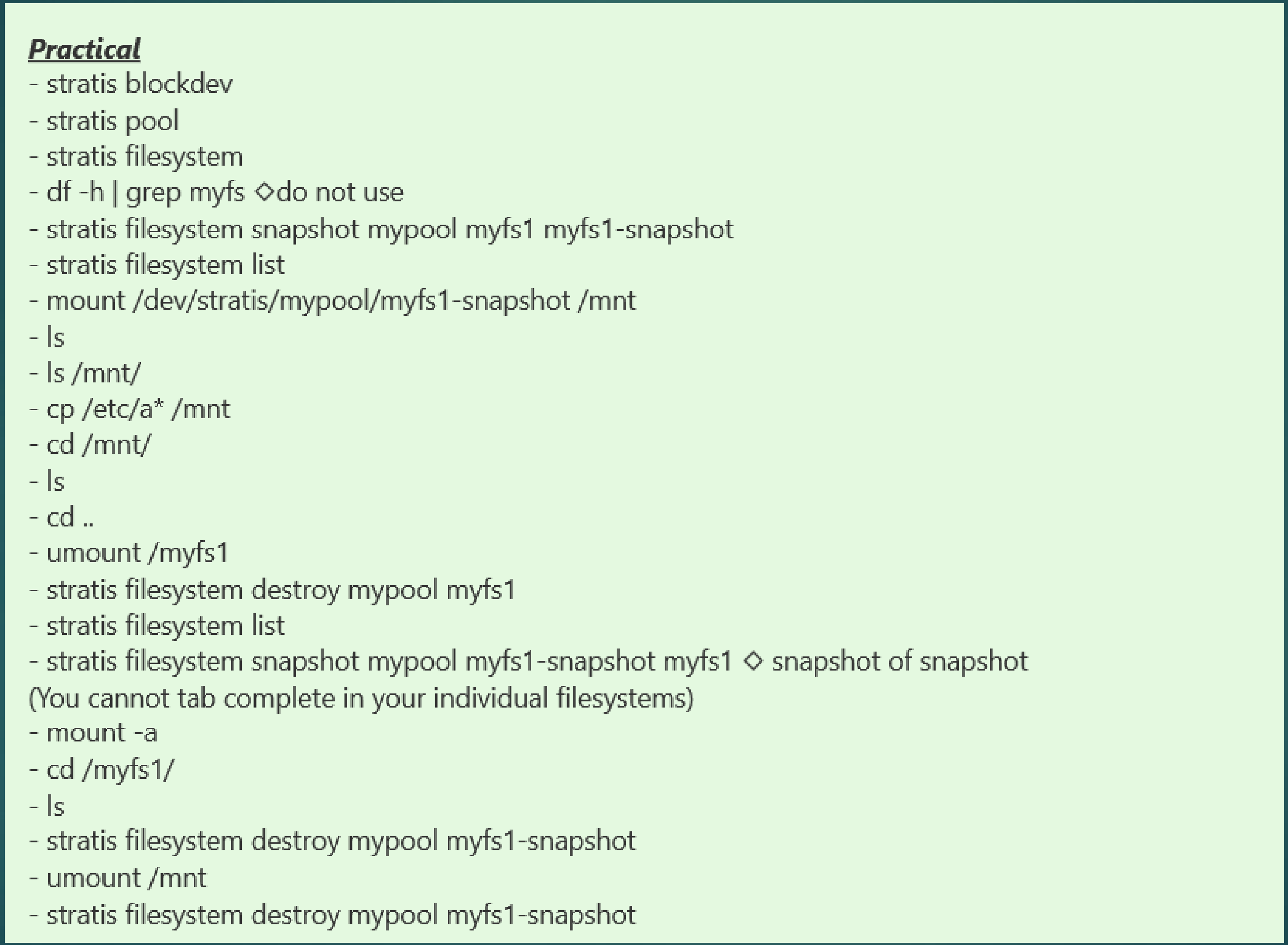
### *Linux 15.8 Managing Stratis Storage features*

- Pools can easily be extended by adding additional block devices
- Use stratis pool add-data nypool /dev/nvme0n3 to add another Block device.

- Standard Linux tools like df don't give accurate sizes as stratis volumes are thin provisioned.
- Use stratis blockdev to show info about all block devices used for stratis.
- Use stratis pool to show info about all pools
- .note that physical used should not come too close to physical size
- Use stratis filesystem to monitor individual filesystems.
- A snapshot is an individual file system that can be mounted.
- After creation snapshots can be modified
- A snapshot and it's origin are not linked : the snapshotted file system can live longer than the file system it was created from
- Each snapshot needs at least half a gigabyte of backing storage for the XFS log(it is a transaction log that logs what is going on your XFS filesystem ).

- stratis fs snapshot mypool myfs1 myfs1-snapshot
- . Changes to the original FS will not be reflected in the snapshot.
- . Use mount /stratis/mypool/my-fs-snapshot /mnt to mount it
- Revert the original volume to the state in the snapshot
- . umount /myfs1
- . stratis fs destroy mypool myfs1
- . stratis fs snapshot mypool myfs1-snap myfs1
- Note that this approach won't work on LVM

- stratis filesystem destroy mypool mysnapshpt will delete a snapshot
- A similar procedure is used for destroying file systems: stratis filesystem destroy mypool myfs
- When there are no more file systems in a pool use stratis pool destroy mypool to delete the pool

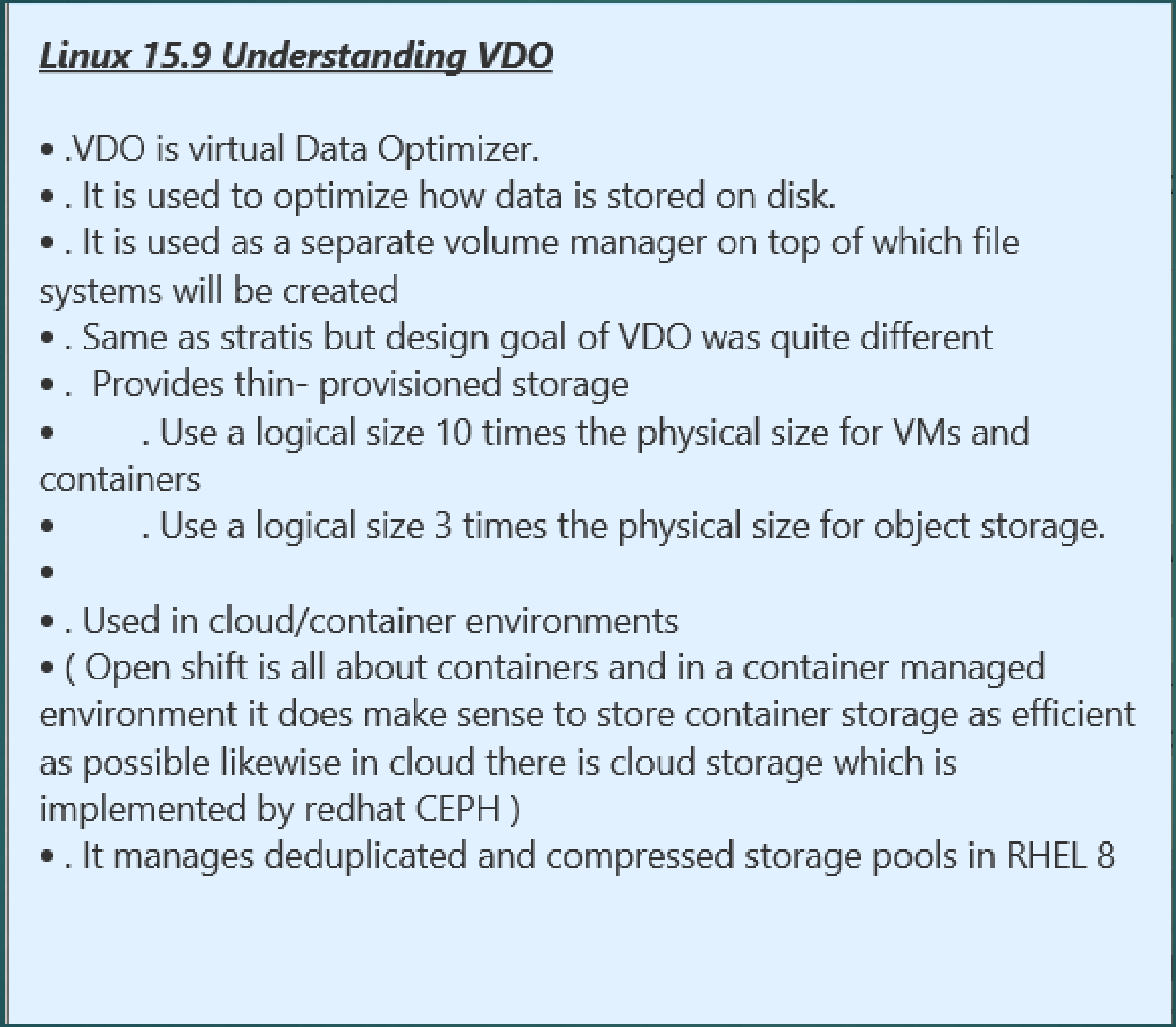


***Practical***

- stratis blockdev
- stratis pool
- stratis filesystem
- df -h | grep myfs ◇do not use
- stratis filesystem snapshot mypool myfs1 myfs1-snapshot
- stratis filesystem list
- mount /dev/stratis/mypool/myfs1-snapshot /mnt
- ls
- ls /mnt/
- cp /etc/a* /mnt
- cd /mnt/
- ls
- cd ..
- umount /myfs1
- stratis filesystem destroy mypool myfs1
- stratis filesystem list
- stratis filesystem snapshot mypool myfs1-snapshot myfs1 ◇ snapshot of snapshot

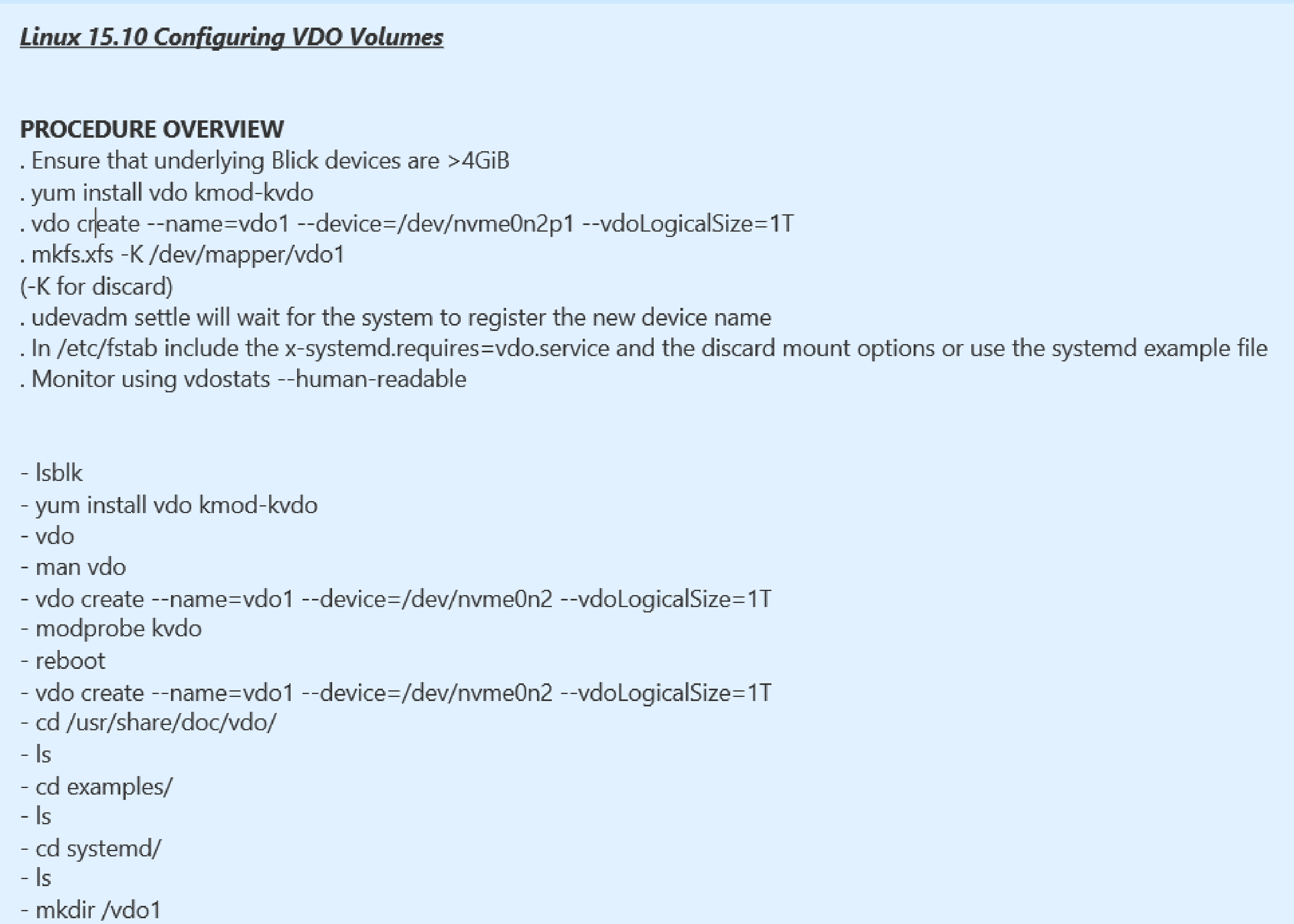
(You cannot tab complete in your individual filesystems)

- mount -a
- cd /myfs1/
- ls
- stratis filesystem destroy mypool myfs1-snapshot
- umount /mnt
- stratis filesystem destroy mypool myfs1-snapshot



## ***Linux 15.9 Understanding VDO***

- .VDO is virtual Data Optimizer.
- . It is used to optimize how data is stored on disk.
- . It is used as a separate volume manager on top of which file systems will be created
- . Same as stratis but design goal of VDO was quite different
- . Provides thin- provisioned storage
- . Use a logical size 10 times the physical size for VMs and containers
- . Use a logical size 3 times the physical size for object storage.
-
- . Used in cloud/container environments
- ( Open shift is all about containers and in a container managed environment it does make sense to store container storage as efficient as possible likewise in cloud there is cloud storage which is implemented by redhat CEPH )
- . It manages deduplicated and compressed storage pools in RHEL 8

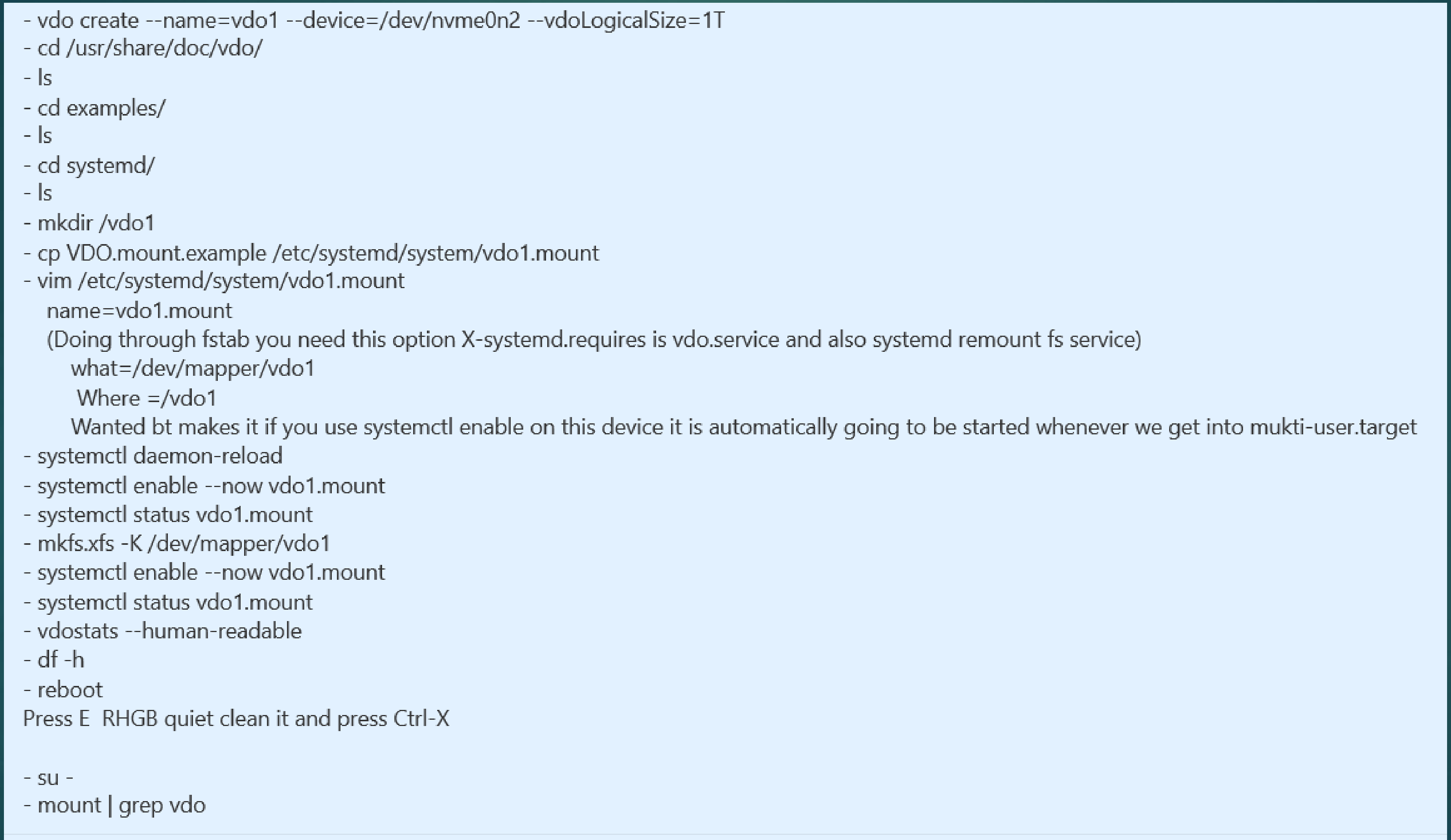


***Linux 15.10 Configuring VDO Volumes***

**PROCEDURE OVERVIEW**

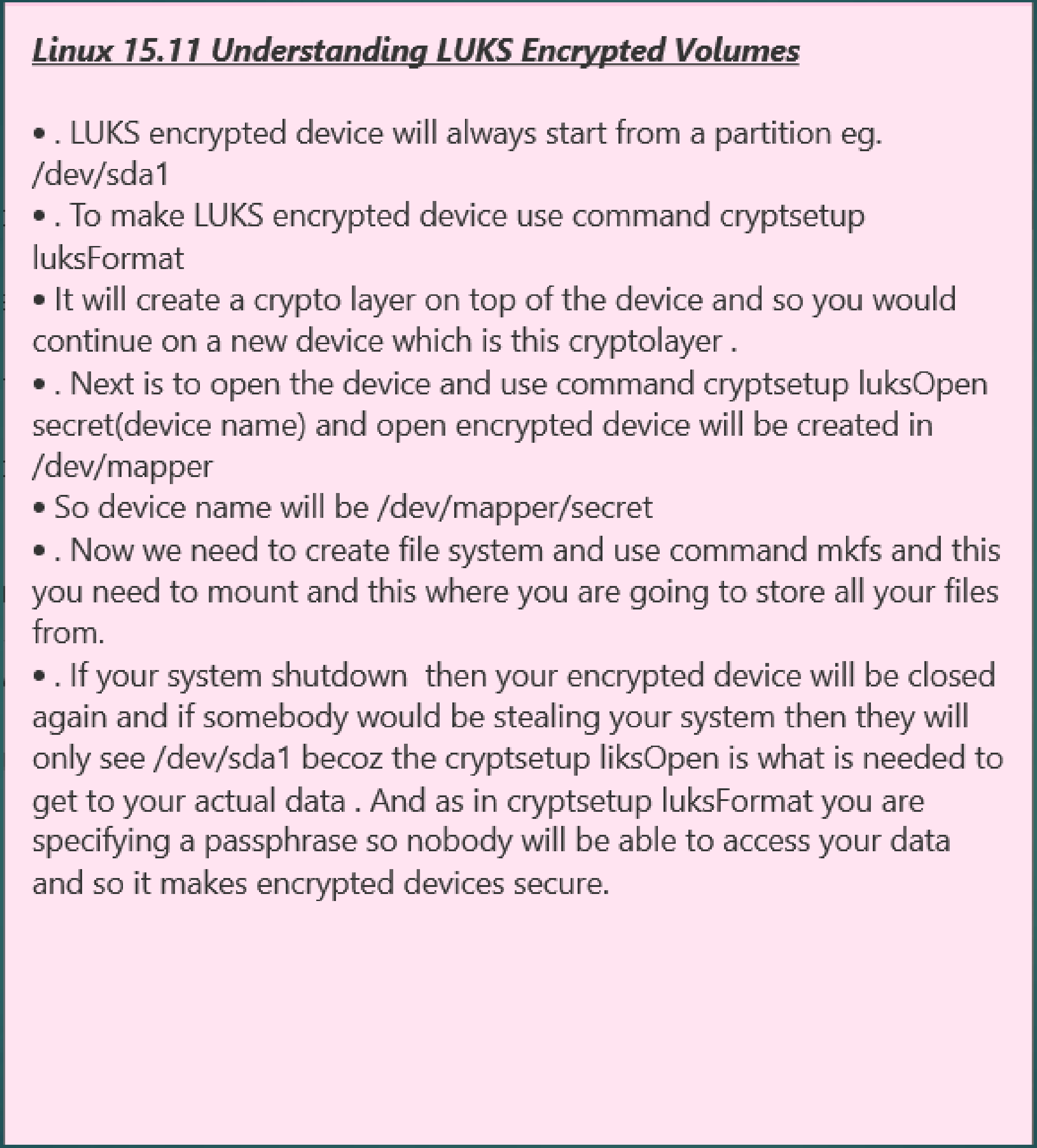
. Ensure that underlying Blick devices are >4GiB
. yum install vdo kmod-kvdo
. vdo create --name=vdo1 --device=/dev/nvme0n2p1 --vdoLogicalSize=1T
. mkfs.xfs -K /dev/mapper/vdo1
(-K for discard)
. udevadm settle will wait for the system to register the new device name
. In /etc/fstab include the x-systemd.requires=vdo.service and the discard mount options or use the systemd example file
. Monitor using vdostats --human-readable

- lsblk
- yum install vdo kmod-kvdo
- vdo
- man vdo
- vdo create --name=vdo1 --device=/dev/nvme0n2 --vdoLogicalSize=1T
- modprobe kvdo
- reboot
- vdo create --name=vdo1 --device=/dev/nvme0n2 --vdoLogicalSize=1T
- cd /usr/share/doc/vdo/
- ls
- cd examples/
- ls
- cd systemd/
- ls
- mkdir /vdo1



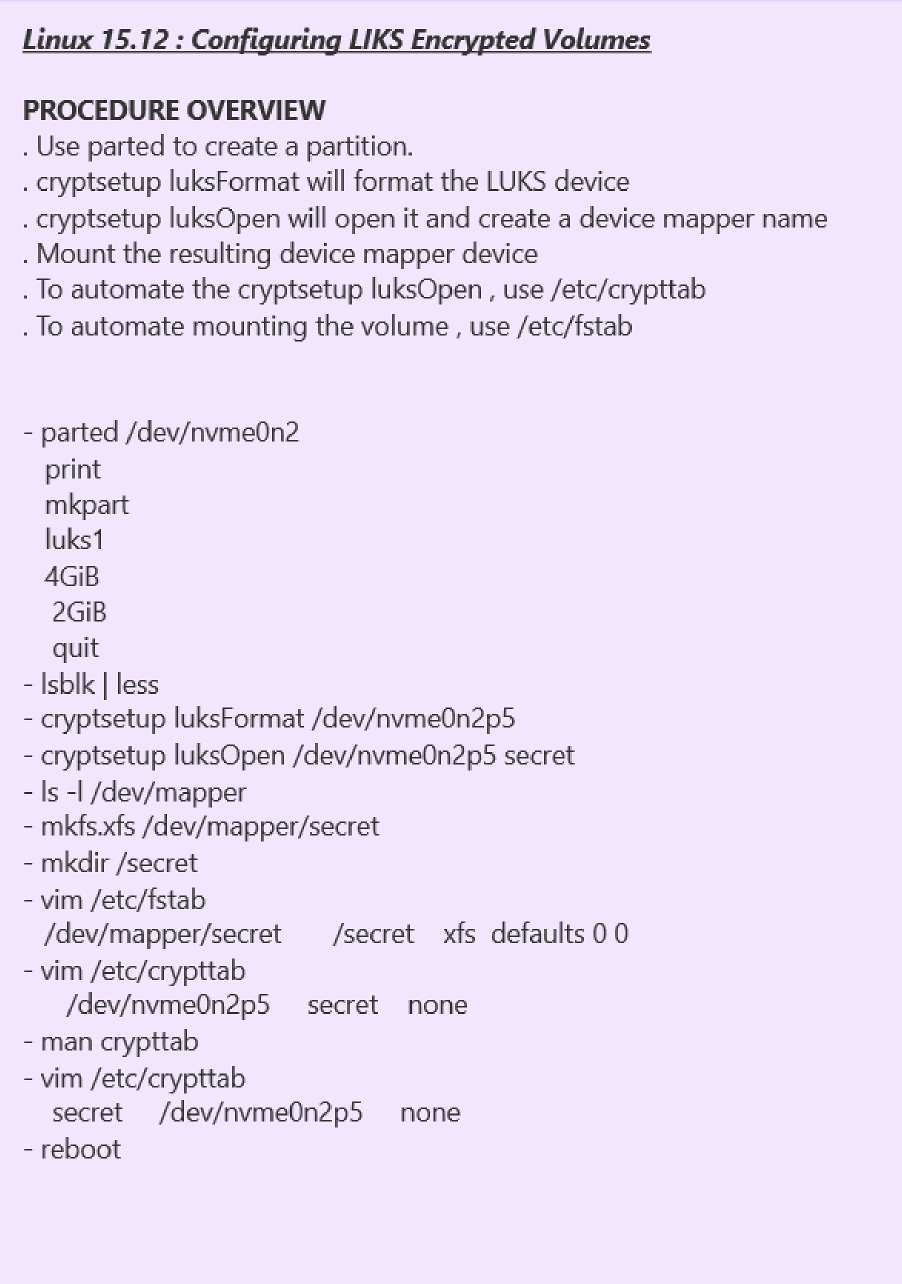
```
- vdo create --name=vdo1 --device=/dev/nvme0n2 --vdoLogicalSize=1T
- cd /usr/share/doc/vdo/
- ls
- cd examples/
- ls
- cd systemd/
- ls
- mkdir /vdo1
- cp VDO.mount.example /etc/systemd/system/vdo1.mount
- vim /etc/systemd/system/vdo1.mount
   name=vdo1.mount
   (Doing through fstab you need this option X-systemd.requires is vdo.service and also systemd remount fs service)
      what=/dev/mapper/vdo1
       Where =/vdo1
      Wanted bt makes it if you use systemctl enable on this device it is automatically going to be started whenever we get into mukti-user.target
- systemctl daemon-reload
- systemctl enable --now vdo1.mount
- systemctl status vdo1.mount
- mkfs.xfs -K /dev/mapper/vdo1
- systemctl enable --now vdo1.mount
- systemctl status vdo1.mount
- vdostats --human-readable
- df -h
- reboot
Press E  RHGB quiet clean it and press Ctrl-X

- su -
- mount | grep vdo
```



***Linux 15.11 Understanding LUKS Encrypted Volumes***

- . LUKS encrypted device will always start from a partition eg. /dev/sda1
- . To make LUKS encrypted device use command cryptsetup luksFormat
- It will create a crypto layer on top of the device and so you would continue on a new device which is this cryptolayer .
- . Next is to open the device and use command cryptsetup luksOpen secret(device name) and open encrypted device will be created in /dev/mapper
- So device name will be /dev/mapper/secret
- . Now we need to create file system and use command mkfs and this you need to mount and this where you are going to store all your files from.
- . If your system shutdown  then your encrypted device will be closed again and if somebody would be stealing your system then they will only see /dev/sda1 becoz the cryptsetup liksOpen is what is needed to get to your actual data . And as in cryptsetup luksFormat you are specifying a passphrase so nobody will be able to access your data and so it makes encrypted devices secure.



## ***Linux 15.12 : Configuring LIKS Encrypted Volumes***

**PROCEDURE OVERVIEW**

. Use parted to create a partition.
. cryptsetup luksFormat will format the LUKS device
. cryptsetup luksOpen will open it and create a device mapper name
. Mount the resulting device mapper device
. To automate the cryptsetup luksOpen , use /etc/crypttab
. To automate mounting the volume , use /etc/fstab

```
- parted /dev/nvme0n2
   print
   mkpart
   luks1
   4GiB
    2GiB
    quit
- lsblk | less
- cryptsetup luksFormat /dev/nvme0n2p5
- cryptsetup luksOpen /dev/nvme0n2p5 secret
- ls -l /dev/mapper
- mkfs.xfs /dev/mapper/secret
- mkdir /secret
- vim /etc/fstab
   /dev/mapper/secret      /secret    xfs  defaults 0 0
- vim /etc/crypttab
      /dev/nvme0n2p5     secret    none
- man crypttab
- vim /etc/crypttab
    secret     /dev/nvme0n2p5     none
- reboot
```

# Thank you